

CLAIMS

What is claimed is:

- 1 1. A method for extending an existing programming language, comprising
2 the steps of:
3 selecting an existing programming language; and,
4 extending an existing programming language by adding at least one
5 language construct defined by a second language.
- 1 2. A method according to claim 1, wherein:
2 said existing programming language is Java.
- 1 3. A method according to claim 1, wherein:
2 said second language is XML.
- 1 4. A method according to claim 1, wherein:
2 said language construct is a parallelism construct representing parallel
3 branch of program execution.
- 1 5. A method according to claim 4, wherein:
2 said parallelism construct further comprises plurality of branch constructs
3 defined by said second language, wherein said branch constructs represent
4 parallel branches of program execution comprising of at least one software
5 activity.
- 1 6. A method according to claim 4, wherein:
2 said parallelism construct is further nested within a similar parallelism
3 construct.
- 1 7. A method according to claim 1, wherein:
2 said language construct is a transaction construct representing
3 transaction block of at least one software activity.

1 8. A method according to claim 7, wherein:
2 said transaction construct further specifies the number of retry attempts to
3 perform the software activities inside said transaction block.

1 9. A method according to claim 7, wherein:
2 said transaction construct is further enclosed within a saga construct
3 comprising of compensation construct with at least one compensating software
4 activity, where in the saga construct represents a long running transaction.

1 10. A method according to claim 9, wherein:
2 said saga construct further comprises of plurality of transaction blocks.

1 11. A method according to claim 1, wherein:
2 said language construct is an exception handlers construct representing
3 an execution mechanism comprising of exception handler construct defined by
4 said second language, which represents exception not caught by the existing
5 programming language handler methods.

1 12. A method according to claim 1, wherein:
2 said language construct is an action construct representing an activity that
3 allows a first software component written using the extended existing
4 programming language to call an operation on a second software component
5 written using said existing programming language.

1 13. A method according to claim 12, wherein:
2 said action construct allows said software component call a piece of Java
3 code.

1 14. A method according to claim 12, wherein:
2 said action construct further allows said second software component call
3 back the said first software component.

1

- 1 15. A method according to claim 1, wherein:
2 said language construct is a multiple receive construct that allows a
3 software component written using the extended existing programming language
4 to wait on multiple input events received.
- 1 16. A method according to claim 15, wherein:
2 said multiple receive construct further allows said software component
3 proceed on a particular branch of program execution, based on the input event
4 that occurred first within the said multiple input events.
- 1 17. A method according to claim 1, wherein:
2 said language construct is a looping construct with ordering of messages
3 received, representing looping functionality, wherein the ordering allows said
4 messages to be received in an order.
- 1 18. A system for extending an existing programming language, comprising:
2 an existing programming language; and,
3 means for extending an existing programming language by adding at least
4 one language construct defined by a second language.
- 1 19. A system according to claim 18, wherein:
2 said existing programming language is Java.
- 1 20. A system according to claim 18, wherein:
2 said second language is XML.
- 1 21. A system according to claim 18, wherein:
2 said language construct is a parallelism construct representing parallel
3 branch of program execution.
- 1 22. A system according to claim 21, wherein:
2 said parallelism construct further comprises plurality of branch constructs
3 defined by said second language, wherein said branch constructs represent
4 parallel branches of program execution comprising of at least one software

5 activity.

1 23. A system according to claim 21, wherein:

2 said parallelism construct is further nested within a similar parallelism
3 construct.

1 24. A system according to claim 18, wherein:

2 said language construct is a transaction construct representing
3 transaction block of at least one software activity.

1 25. A system according to claim 24, wherein:

2 said transaction construct further specifies the number of retry attempts to
3 perform the software activities inside said transaction block.

1 26. A system according to claim 24, wherein:

2 said transaction construct is further enclosed within a saga construct
3 comprising of compensation construct with at least one compensating software
4 activity, where in the saga construct represents a long running transaction.

1 27. A system according to claim 26, wherein:

2 said saga construct further comprises of plurality of transaction blocks.

1 28. A system according to claim 18, wherein:

2 said language construct is an exception handlers construct representing
3 an execution mechanism comprising of exception handler construct defined by
4 said second language, which represents exception not caught by the existing
5 programming language handler methods.

1 29. A system according to claim 18, wherein:

2 said language construct is an action construct representing an activity that
3 allows a first software component written using the extended existing
4 programming language to call an operation on a second software component
5 written using said existing programming language.

1 30. A system according to claim 29, wherein:
2 said action construct allows said software component call a piece of Java
3 code.

1 31. A system according to claim 29, wherein:
2 said action construct further allows said second software component call
3 back the said first software component.

1 32. A system according to claim 18, wherein:
2 said language construct is a multiple receive construct that allows a
3 software component written using the extended existing programming language
4 to wait on multiple input events received.

1 33. A system according to claim 32, wherein:
2 said multiple receive construct further allows said software component
3 proceed on a particular branch of program execution, based on the input event
4 that occurred first within the said multiple input events.

1 34. A system according to claim 18, wherein:
2 said language construct is a looping construct with ordering of messages
3 received, representing looping functionality, wherein the ordering allows said
4 messages to be received in an order.

1 35. A computer system comprising:
2 a processor;
3 object code executed by said processor, said object code configured to:
4 extend an existing programming language by adding a
5 language construct defined by a second language.

1 36. A method for extending Java programming language, comprising the
2 steps of:
3 selecting Java programming language; and,
4 extending Java programming language by adding at least one language

5 construct defined by XML.

1 37. A system for extending Java programming language, comprising:
2 a Java programming language; and,
3 means for extending Java programming language by adding at least one
4 language construct defined by XML.

1 38. A method for creating a program, comprising the steps of:
2 selecting an existing programming language extended with at least one
3 language construct defined by a second language; and
4 creating a program using the extended existing programming language.

1 39. A computer program product, comprising:
2 a program created by using an existing programming
3 language extended with at least one language construct defined by a second
4 language.